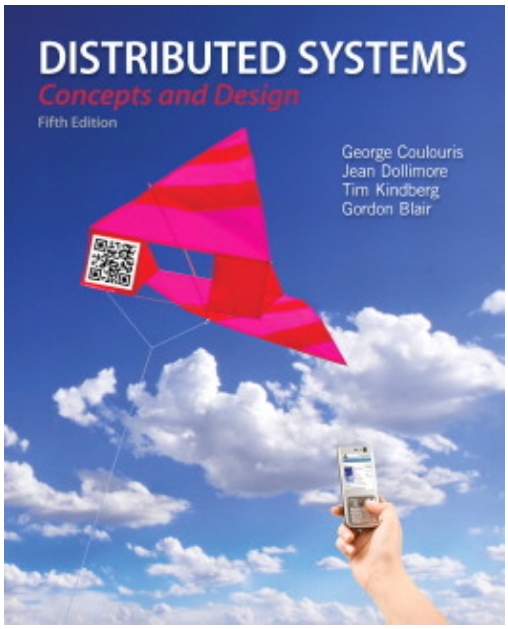
Distributed Systems
COMP90015 2017 SM1
Indirect Communication

Indirect Communication

From Coulouris, Dollimore and Kindberg, *Distributed Systems: Concepts and Design*, Edition 5, © Addison-Wesley 2012.



Lectures prepared by: Aaron Harwood

Overview

- Group Communication
- Publish/Subscribe
- Message Queues
- Shared Memory

Introduction

Indirect communication is defined as communication between entities in a distributed system through an intermediary with no direct coupling between the sender and the receiver(s).

- *Space uncoupling*: sender does not know or need to know the identity of the receiver(s)
- *Time uncoupling*: sendder and receiver can have independent lifetimes, they do not need to exist at the same time

Time uncoupling is not synonomous with asynchronous communication. Asynchronous communication doesn't imply that the receiver has an independent lifetime, in otherwords we could consider a time coupled asynchronous system.

Group Communication

Group communication offers a space uncoupled service whereby a message is sent to a group and then this message is delivered to all members of the group. It provides more than a primitive IP multicast:

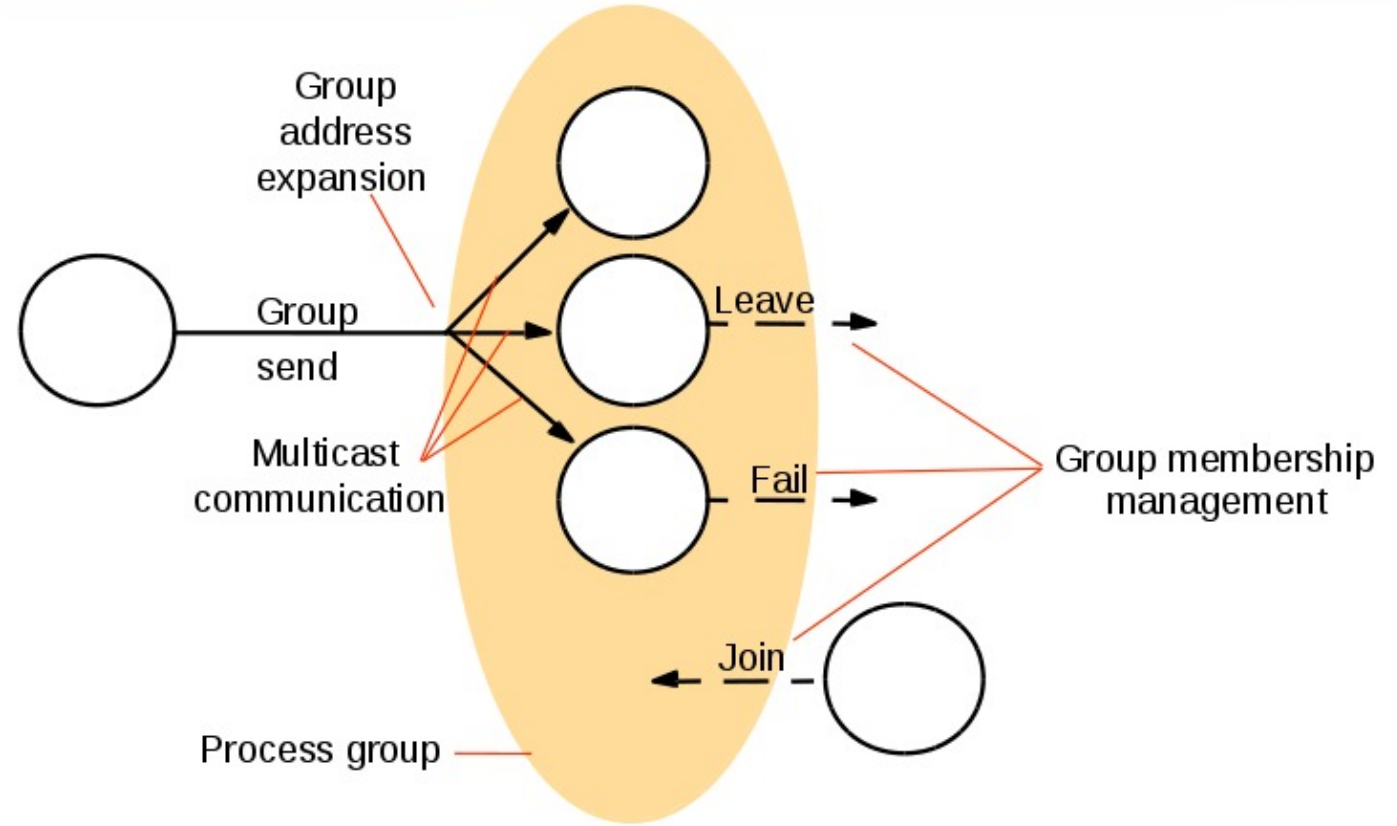
- manages group membership
- detects failures and provides reliability and ordering guarantees

Typical aspects of an API:

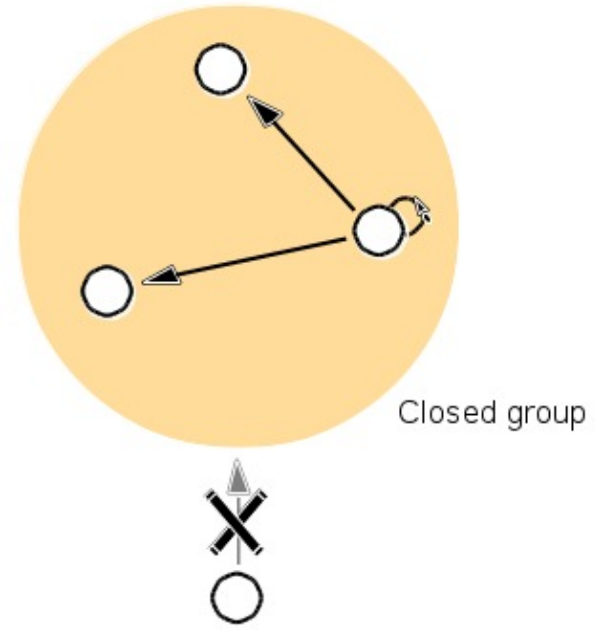
- group
- group membership
- join
- leave
- multicast, broadcast

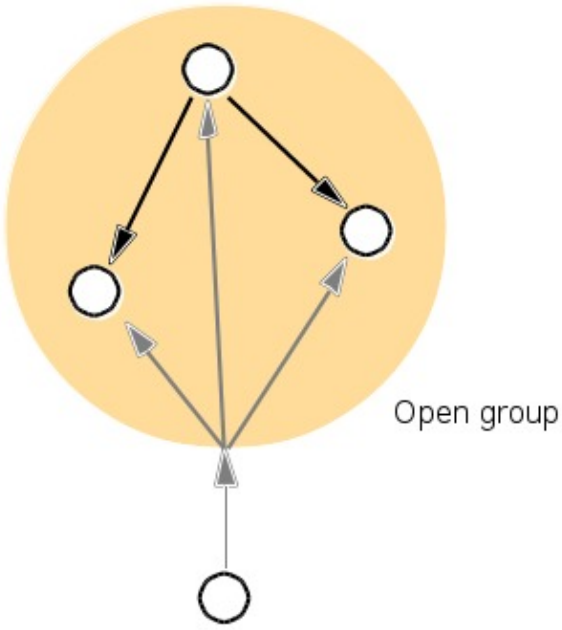
Efficient sending to multiple receivers, instead of multiple independent send operations, is an essential feature of group communication.

Group Model



Group services





- closed groups only allow group members to multicast to it
- overlapping groups allows entities to be members of multiple groups
- synchronous and asynchronous variations can be considered

Implementation issues

- reliability and odering in multicast
  - FIFO (first in first out) ordering is concerned with preserving the order from the prespective of a sender process
  - causal ordering, a message that *happens before* another message will be preserved in that order in the delivery at all processes
  - total ordering, if a message is delivered before another message at one process then this is preserved at all processes
- group membership management
  - group members leave and join
  - failed members
  - notifying members of group membership changes
  - changes to the group address

Publish/Subscribe Systems

Publish/subscribe systems are sometimes referred to as *distributed event-based systems*.

A publish/subscribe system is a system where *publishers* publish structured events to an event service and *subscribers* express interest in particular events through *subscriptions* which can be arbitrary patterns over the structured events.

- financial information systems
- live feeds of real-time data, e.g. RSS feeds
- support for cooperative working, where a number of participants need to be informed of events of shared interest
- support for ubiquitous computing, including management of events emanating from the ubiquitous infrastructure, e.g. location events
- a broad set of monitoring applications, including network monitoring in the Internet

Events and notifications

RMI and RPC support the synchronous communication model where the client invoking the call waits for the results to be returned.

Events and *notifications* are associated with the asynchronous communication model.

Distributed event-based systems can use the *publish-subscribe* communication paradigm:

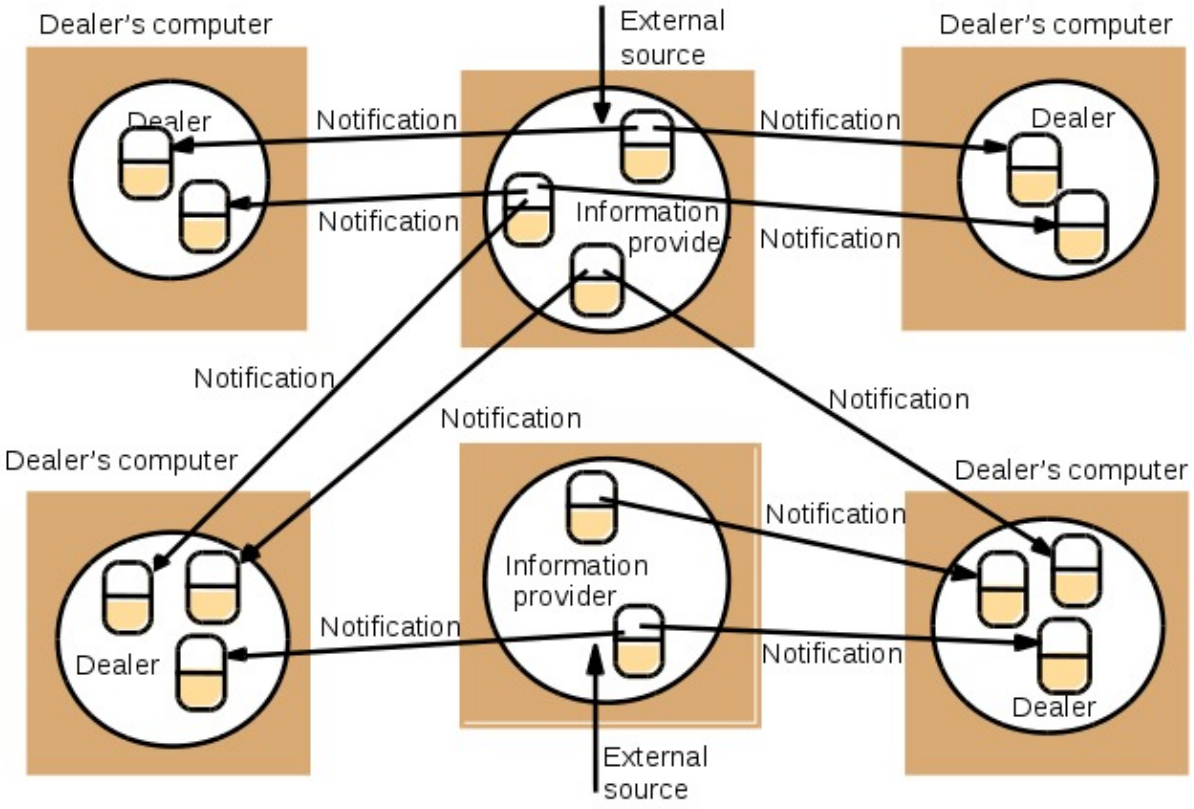
- Objects that generate events *publish* information that are of interest to other objects.
- Objects that are interested in a particular type of event *subscribe* to the type of events.
- Publishers and subscriber are loosely coupled.

Characteristics of distributed event-based systems

Heterogeneity: Allows objects that were not designed to interoperate to communicate due to the loosely coupled nature.

Asynchronous: Communication is asynchronous and event driven.

Example: Simple dealing room system



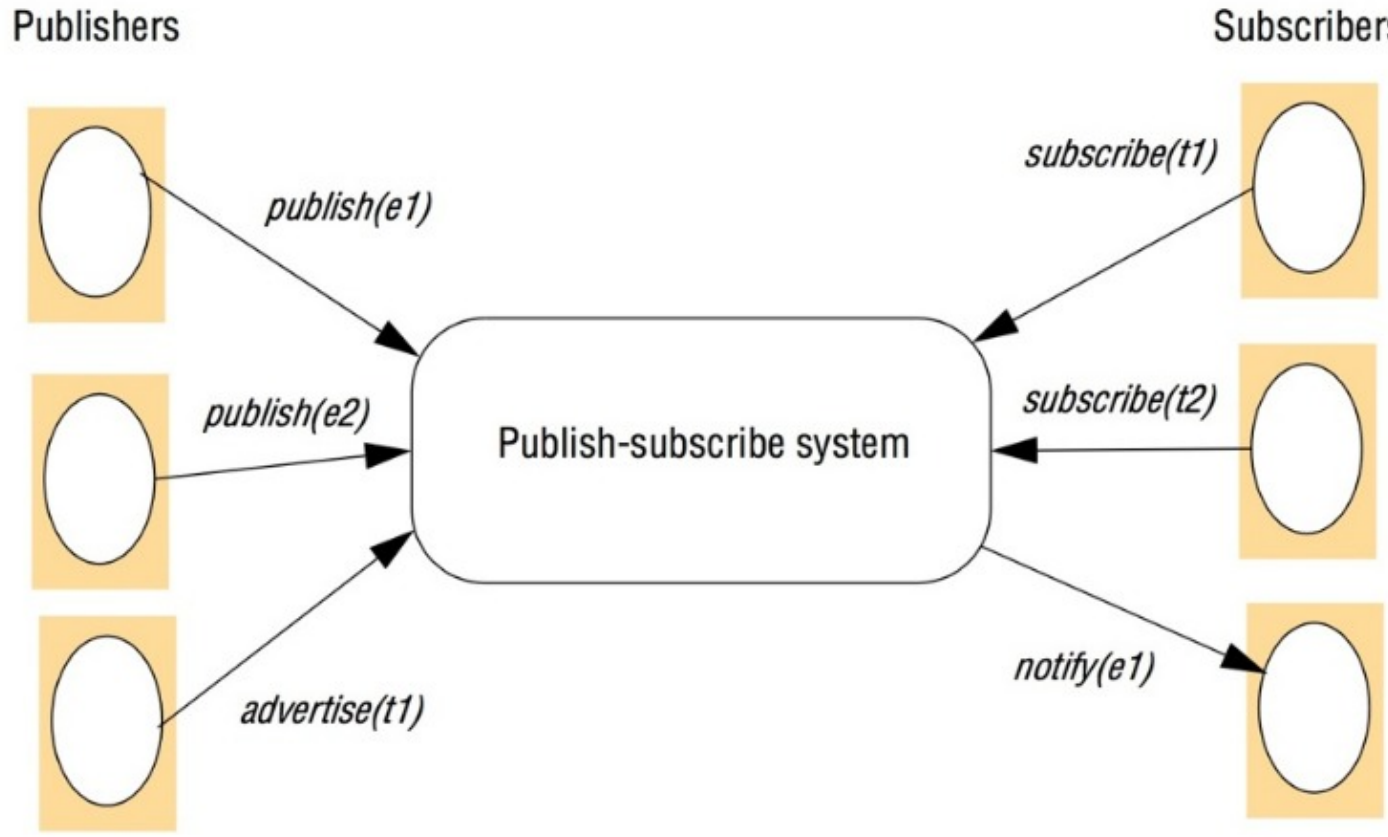
Event Types

Events sources can generate different *types* of events. *Attributes* contain informations about the event.

Types and attributes are used by subscribers when subscribing to events.

Notifications occur when event types and attributes match to that of subscriptions.

Programming model



Advertise provides an additional mechanism for publishers to declare the nature of future events, i.e. the types of events of interest that may occur.

Types of publish-subscribe systems

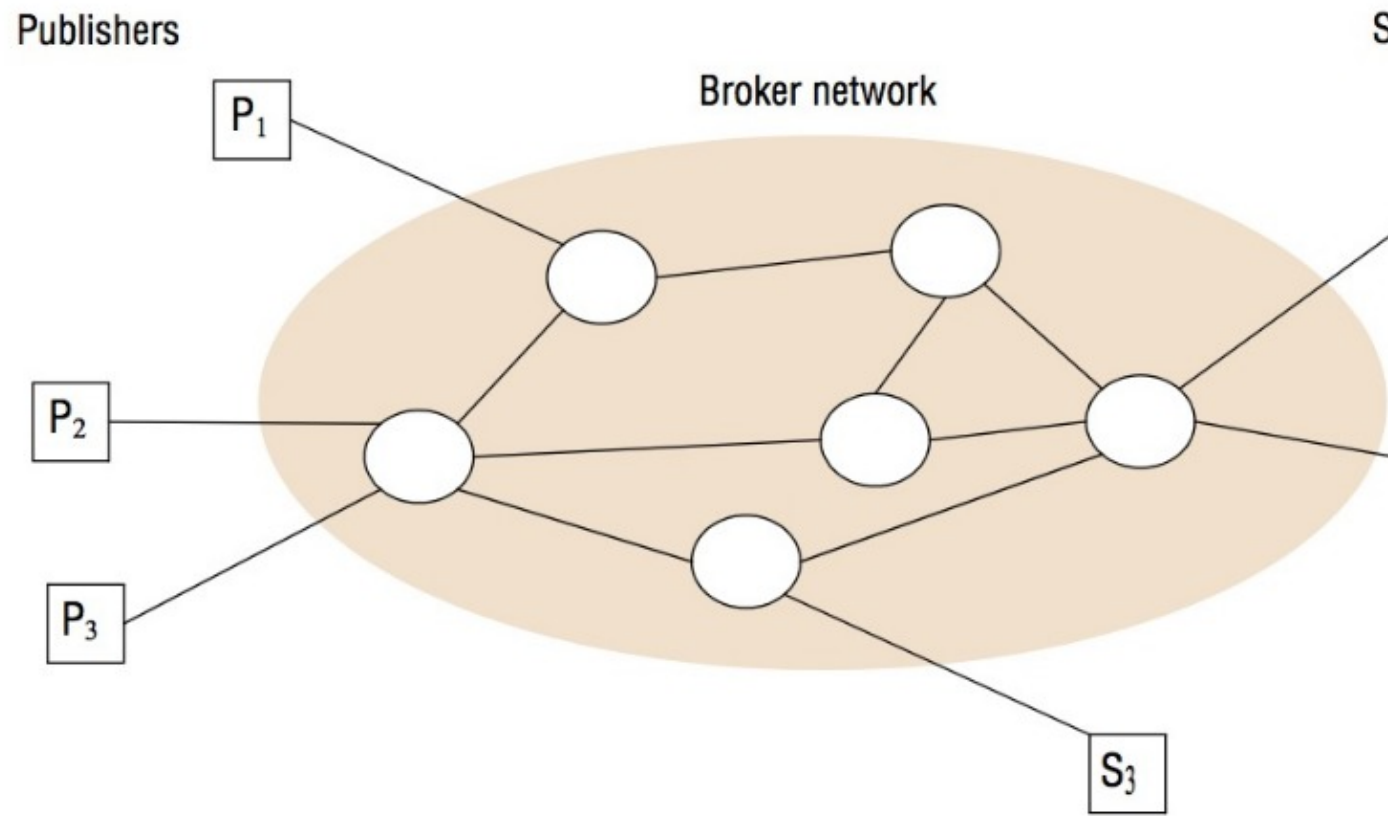
Channel Based: Publishers publish to named channels and subscribers subscribe to all events on a named channel.

Type Based: Subscribers register interest in types of events and notifications occur when particular types of events occur.

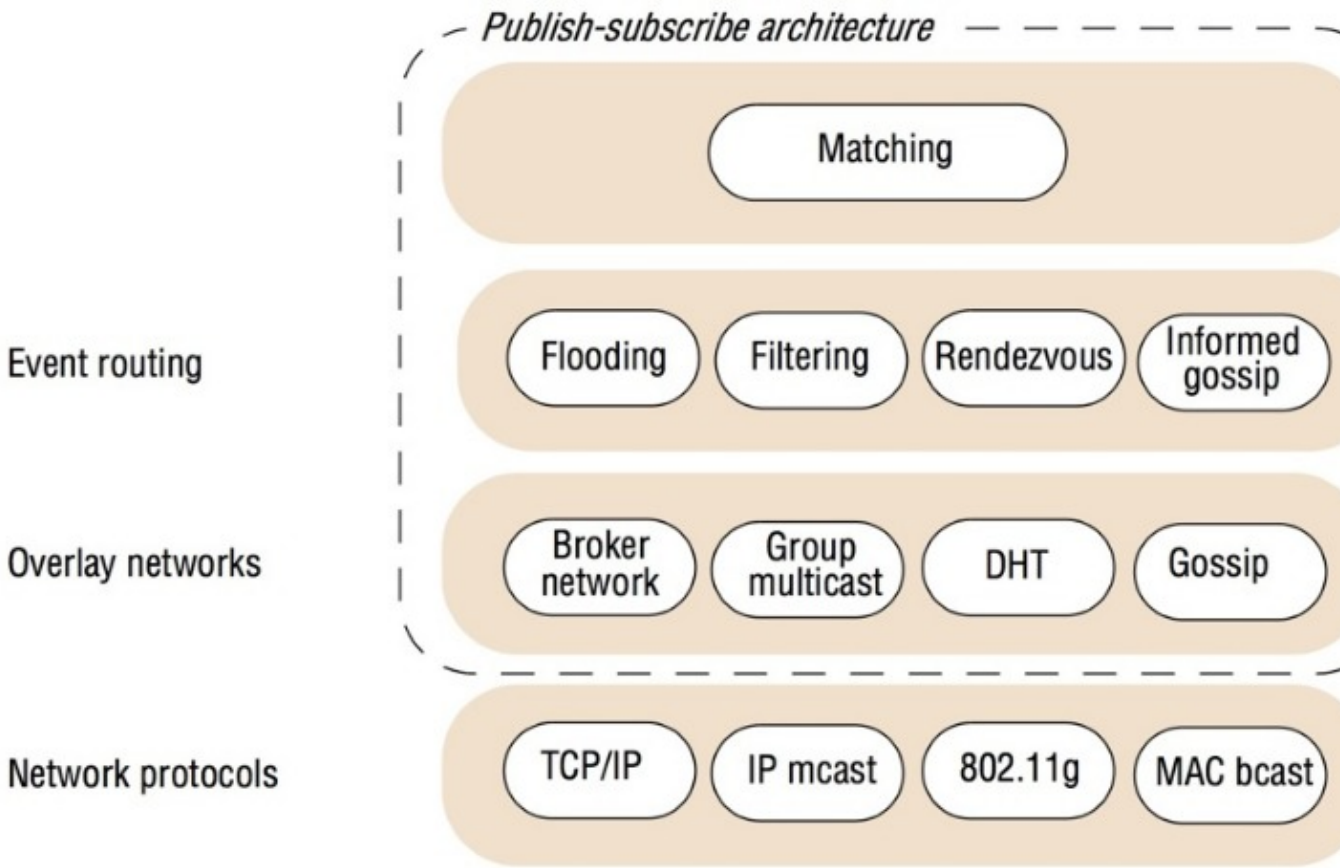
Topic Based: Subscribers register interest in particular topics and notifications occur when any information related to the topic arrives.

Content Based: This is the most flexible of the schemes. Subscribers can specify interest is particular values or ranges of values for multiple attributes. Notifications are based on matching the attribute specification criteria.

Centralized versus decentralized



Overall System Architecture

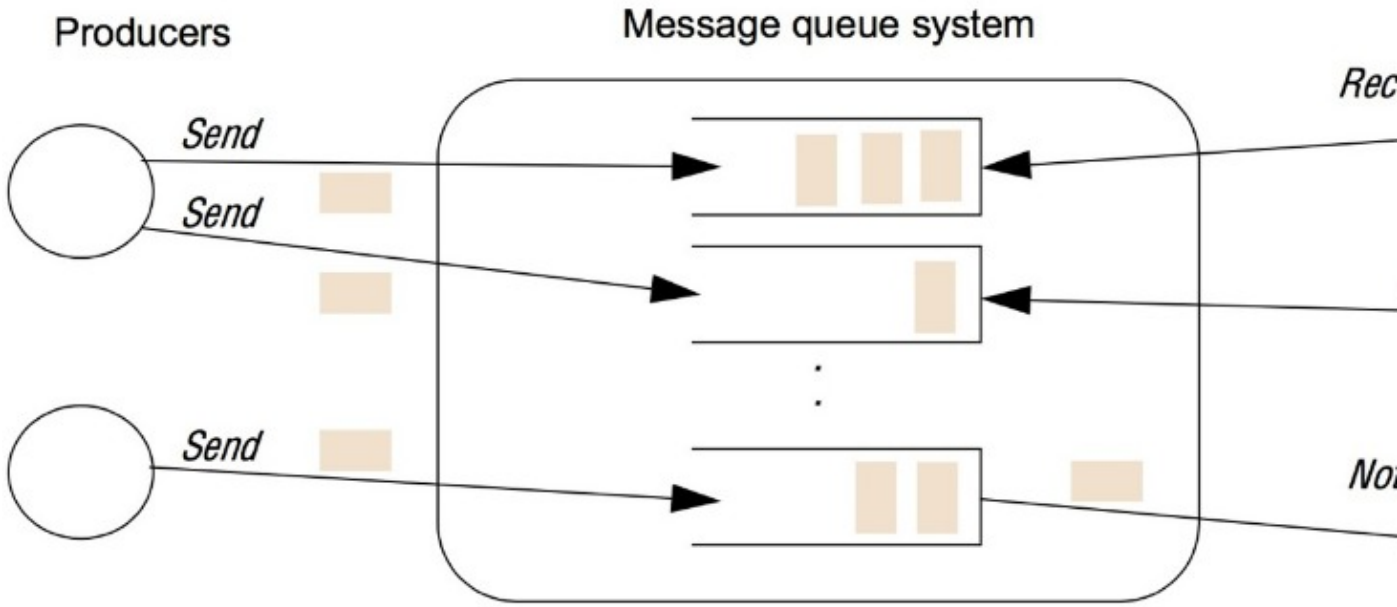


Examples of pub/sub systems

System (and further reading)	*Subscription model*	*Distribution model*	
CORBA Event Service (Chapter 8)	Channel-based	Centralized	-
TIB Rendezvouz [Oki *et al.* 1993]	Topic-based	Distributed	Ff
Scribe [Castro *et al.* 2002b]	Topic-based	Peer-to-peer (DHT)	Re
TERA [Baldoni *et al.* 2007]	Topic-based	Peer-to-peer	In
Siena [Carzaniga *et al.* 2001]	Content-based	Distributed	Fi
Gryphon [www.research.ibm.com]	Content-based	Distributed	Fi
Hermes [Pietzuch and Bacon 2002]	Topic- and content-based	Distributed	Re fil
MEDYM [Cao and Singh 2005]	Content-based	Distributed	Fl
Meghdoot [Gupta *et al.* 2004]	Content-based	Peer-to-peer	Re
Structure-less CBR [Baldoni *et al.* 2005]	Content-based	Peer-to-peer	In

Message Queues

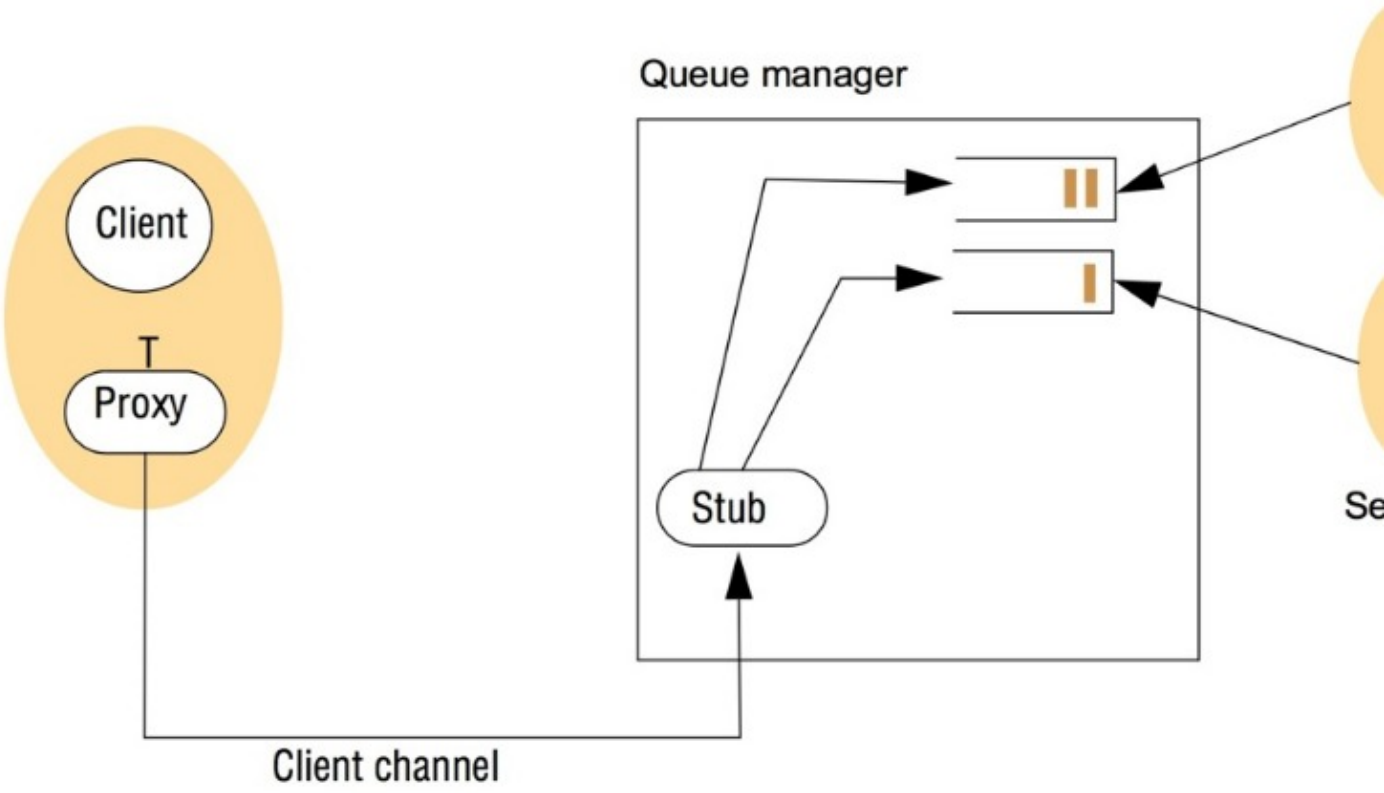
Whereas groups and publish/subscribe provide a one-to-many style of communication, message queues provide a point-to-point service using the concept of a message queue as an indirection, thus achieving the desired properties of space and time uncoupling. The are point-to-point in that the sender places the message into a queue, and it is then removed by a single process.



Programming model

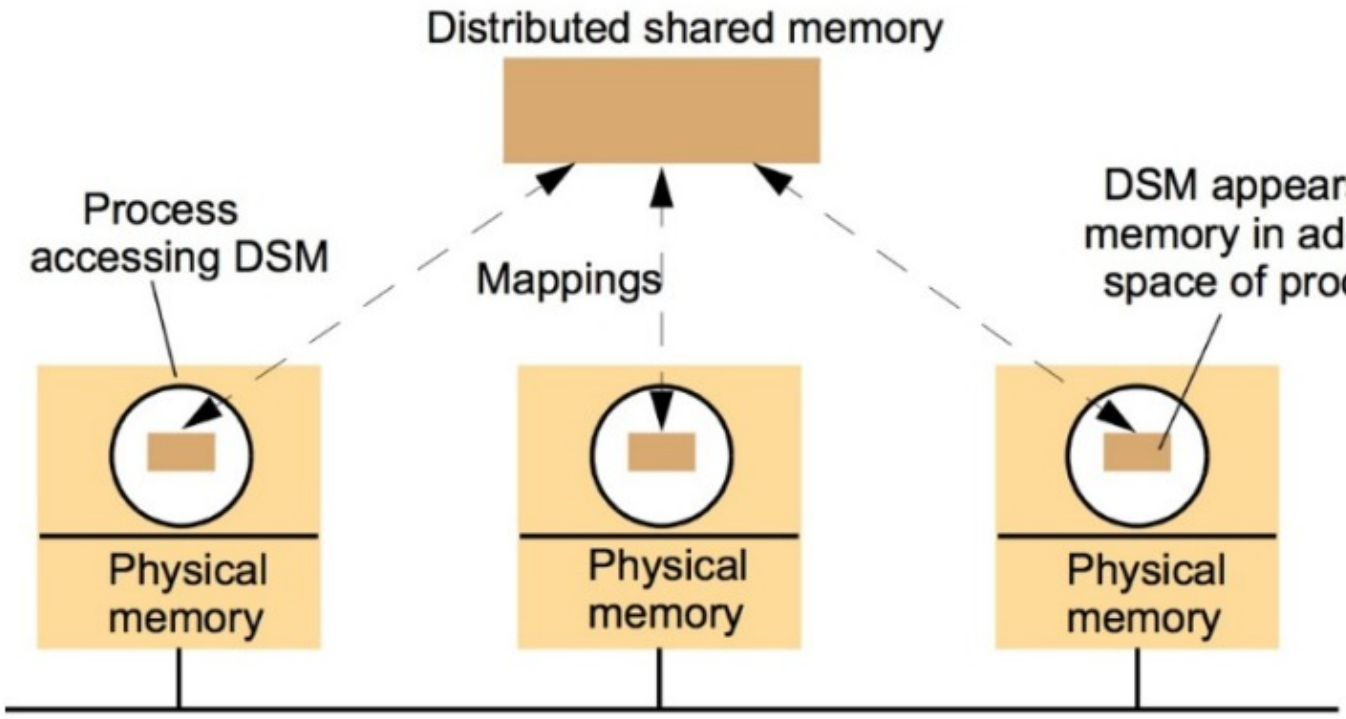
- **send**: producers put a message on a particular queue
- **blocking receive**: a consumer waits for at least one message on a queue then returns
- **non-blcoking receive**: or poll, a consumer will check and get a message if there, otherwise it returns without a message
- **notify**: an event is generated at the receiver when a message is available

Example WebSphere MQ



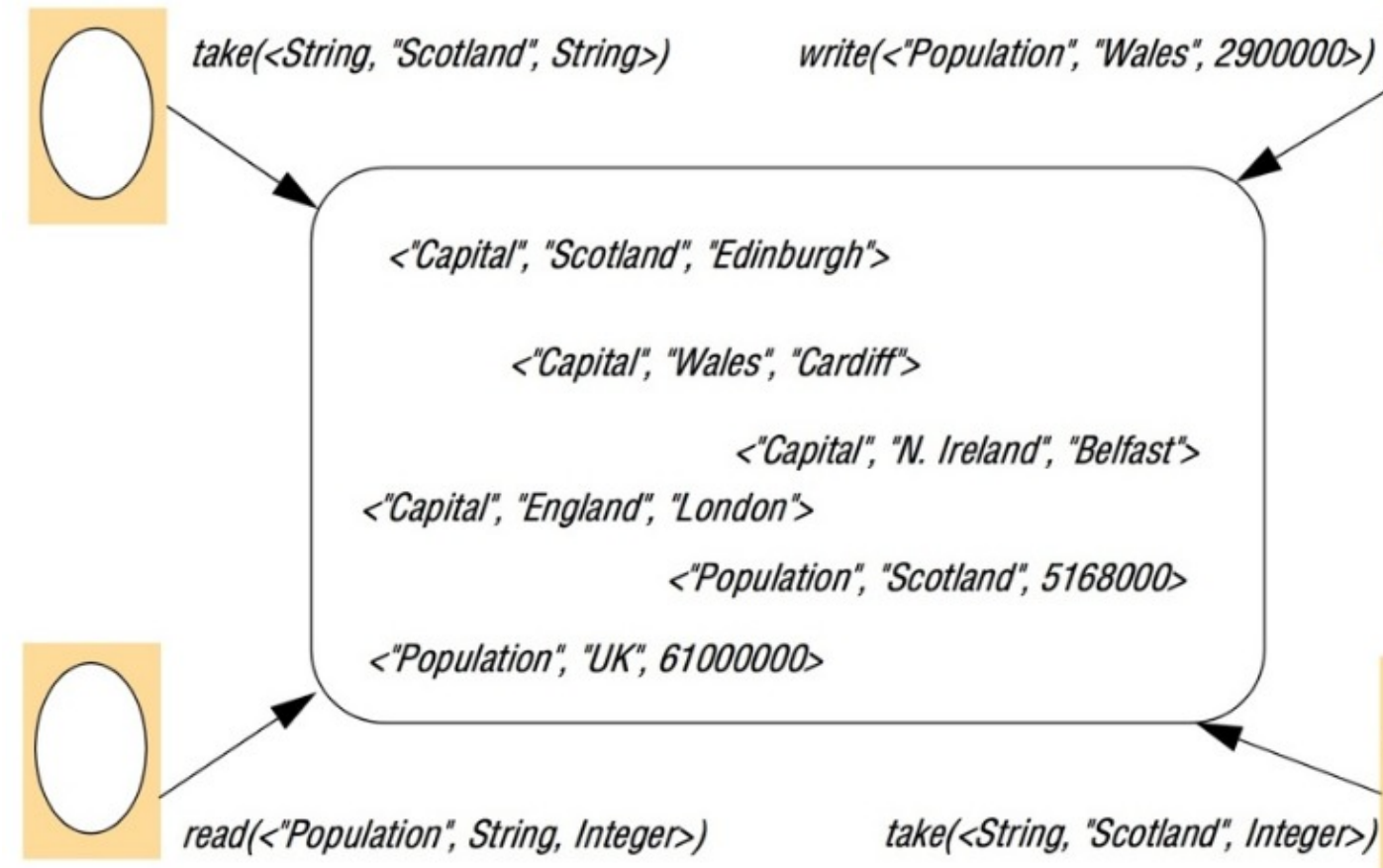
Shared memory approaches

Distributed shared memory is an abstraction for sharing data between computers that do not share physical memory. Processes access DSM by reads and updates to what appears to be ordinary memory within their address space.



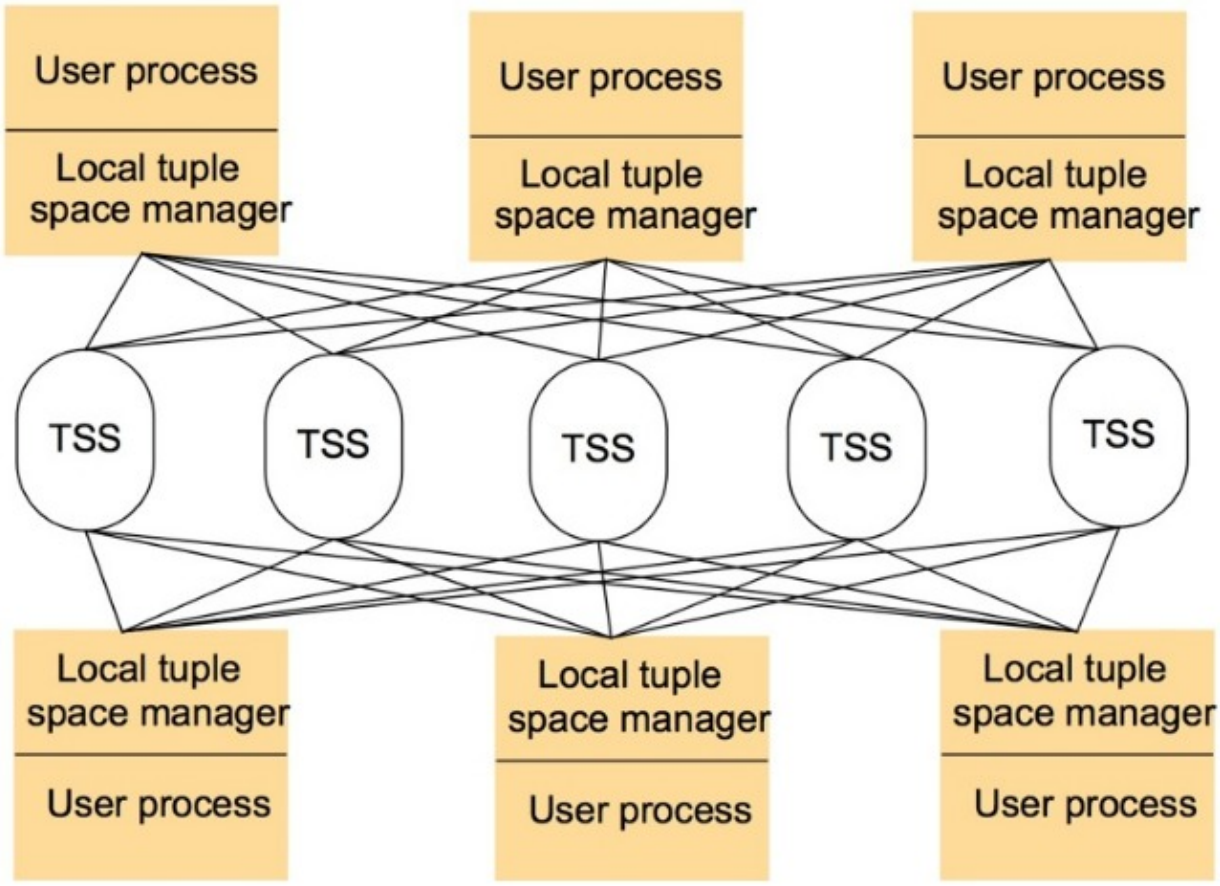
Tuple Spaces

The tuple space is a more abstract form of shared memory, compared to DSM.



Example York Linda Kernel

The implementation uses multiple Tuple Space Servers.


User process
Local tuple space manager
User process
Local tuple space manager
User process
Local tuple space manager
TSS
TSS
TSS
TSS
TSS
Local tuple space manager
User process
Local tuple space manager
User process
Local tuple space manager
User process

Summary

	Groups	*Publish-subscribe systems*	*Message queues*	*DSM*	*Tuple spaces*
Space-uncoupled	Yes	Yes	Yes	Yes	Yes
Time-uncoupled	Possible	Possible	Yes	Yes	Yes
Style of service	Communication-based	Communication-based	Communication-based	State-based	State-based
Communication pattern	1-to-many	1-to-many	1-to-1	1-to-many	1-1 or 1-to-man
Main intent	Reliable distributed computing	Information dissemination or EAI; mobile and ubiquitous systems	Information dissemination or EAI; commercial transaction processing	Parallel and distributed computation	Parallel and distributed computation; mobile and ubiquitous systems
Scalability	Limited	Possible	Possible	Limited	Limited
Associative	No	Content-based publish-subscribe only	No	No	Yes